

TRANSFORMATIONAL CONVERSATION DEFINITION LANGUAGE

1 **Technical Field**

2 The present invention generally relates to computer languages and, more
3 particularly to a computer language for facilitating communication between web services,
4 where web services may use disparate input and output document formats.

5 **Background**

6 Distributed computing has evolved from intra-enterprise application integration,
7 where application developers work together to develop and code to agreed upon method
8 interfaces, to inter-enterprise integration, where E-Services may be developed by
9 independent enterprises with completely disjoint computing infrastructures. To
10 accommodate the change, E-Services, which may include services, agents, and web-
11 services, should be able to communicate and exchange business data in a meaningful
12 way, and have some degree of flexibility and autonomy with regard to the interactions.

13 Several existing agent systems allow services/agents to communicate following
14 conversational protocols. However, all of these agent systems are tightly coupled to
15 specific service/agent systems, and require that all participating entities be built upon a
16 common service/agent platform.

17 Peer-to-Peer (P2P) technology brings distributed computing capabilities to
18 individuals, creating a new perspective of network-capable devices in the role of
19 resources that can be combined to enable new capabilities greater than the sum of the
20 parts. As services become more loosely coupled and increasingly autonomous,
21 heterogeneous distributed services should be able to discover and converse with each
22 other dynamically, with or without human intervention. Current paradigms of service
23 interaction, however, require service developers to hardcode their logic to adhere strictly
24 to pre-defined conversation policies.

25 For example, Bradshaw, J.M. provides an open distributed architecture for
26 software agents, e.g., the Knowledgeable Agent-oriented System (KAoS), in the 1996
27 issue of “Knowledge Acquisition for Knowledge-Based Systems Workshop,” entitled
28 “KAoS: An Open Agent Architecture Supporting Reuse, Interoperability, and
29 Extensibility.” However, as with other conventional techniques, KAoS requires agent
30 developers to hard-wire conversation policies into agents in advance.

1 Walker, A. and Wooldridge, M. address the issue of how a group of autonomous
2 agents can reach a global agreement on conversation policy in the 1995 issue of "First
3 International Conference on Multi-Agent Systems," entitled "Understanding The
4 Emergence Of Conventions In Multi-Agent Systems." However, Walker and Wooldridge
5 require the agents themselves to implement strategies and control.

6 Chen, Q., Dayal, U., Hsu, M., and Griss, M. provide a framework in which agents
7 can dynamically load conversation policies from one-another in the 2000 issue of "First
8 International Conference on E-Commerce and Web-Technology," entitled "Dynamic
9 Agents, Workflow and XML for E-Commerce Automation." But the solution of Chen et
10 al. is homogeneous and requires that agents be built upon a common infrastructure.

11 A few E-Commerce systems also support conversations between services.
12 However, these systems all require that the client and service developers implement
13 matching conversation control policies.

14 For example, RosettaNet's Partner Interface Processes (PIPs) specify roles and
15 required interactions between two businesses, while Commerce XML (cXML) is a
16 proposed standard being developed by more than 50 companies for business-to-business
17 electronic commerce. However, both RosettaNet and CommerceOne require that
18 participating services pre-conform to their standards.

19 To illustrate, in an E-Service marketplace with two different enterprises, a client
20 service in one enterprise may have discovered a storefront service in another enterprise.
21 In order to complete a sale, a credit validation service in yet another enterprise may be
22 employed by the storefront service to make sure that the client is credible. These services
23 can communicate by exchanging messages using common transports and message
24 formats. The storefront service may expect that message exchanges, i.e., the
25 conversation, follow a specific pattern. So does the credit validation service. Because the
26 client and the storefront services belong to different enterprises and have discovered each
27 other dynamically, the client service may not know what conversations the storefront
28 service supports. Similarly, the credit validation service may not know what
29 conversations the client service or the storefront service supports. Accordingly, explicit
30 conversation control implementation may be needed to conduct a conversation between
31 the client service and the storefront service, between the client service and the credit
32 validation service, and between the storefront service and the credit validation service.

33 Accordingly, current conversation systems require participating service
34 developers to implement logic code to adhere strictly to pre-defined conversation policies.

1 Should a conversation protocol change, all participating services that support the protocol
2 must be updated and recompiled, reducing the likelihood that two services that discover
3 each other will be able to converse spontaneously.

4 WSCL (Web Services Conversation Language) addresses the problem of how to
5 enable E-Services from different enterprises to engage in flexible and autonomous, yet
6 potentially quite complex, business interactions. It adopts an approach from the domain
7 of software agents, modeling protocols for business interaction as conversation policies,
8 but extends this approach to exploit the fact that E-Service messages are typically XML-
9 based business documents and can thus be mapped to XML document types. Each
10 WSCL specification describes a single type of conversation from the perspective of a
11 single participant. A service can participate in multiple types of conversations.
12 Furthermore, a service can engage in multiple simultaneous instances of a given type of
13 conversation. However, WSCL does not include any means of specifying document
14 transformations.

15 There are other, similar, business interaction languages, but none of these include
16 document transformations. For instance, the Oracle Integration Server, as described by C.
17 Bussler, "Semantic B2B Integration Server Technology as Infrastructure for Electronic
18 Hubs," *First International Workshop on Electronic Business Hubs: XML, Metadata,*
19 *Ontologies, and Business Knowledge on the Web* (September 2001), includes a semantic
20 transformation engine for transforming documents, but requires the application
21 developers to provide explicit document transformations to an intermediary form specific
22 to the integration server.

23 Further, the problem of disparate services utilizing different conversation
24 protocols (flow) is exacerbated by similar conversation protocols using documents that
25 vary insignificantly. A mere change in a field name can thwart a program from
26 autonomously managing a conversation between two services. It is therefore desirable to
27 enable automatic transformation of documents during a conversation.

28 **Summary**

29 It is an aspect of an embodiment of the invention to allow the implementation of
30 systems that are decentralized and conducive to dynamic and autonomous interactions
31 between applications that have been independently developed with a minimum of
32 coordination. In particular, an aspect of an embodiment of the invention addresses the
33 problem of how to enable services that support conflicting message document types to

1 interact in a meaningful way without service developers having to implement any
2 document transformation logic.

3 A preferred embodiment of the present invention associates document
4 transformations with conversation-specific service interactions. Specifically, a
5 conversation definition language is extended with document transformational elements or
6 specifications, creating a Transformational Conversation Definition Language (TCDL).

7 In one embodiment, the TCDL description includes sections for document type
8 description defining all inbound and outbound documents used in the conversation. The
9 TCDL also describes interactions of the conversation, or rather, models the states of the
10 conversation. A transition section is included which defines the flow of the conversation,
11 or rather, the temporal syntax. The interactions defined in the TCDL are extended with
12 transformation definitions to accommodate varying document formats used by different
13 services.

14 A conversation controller acts as a go-between for two or more services.
15 Documents are passed back and forth between two services during a conversation. In one
16 embodiment, a document requiring transformations is transformed by the conversation
17 controller before being either sent or received by the service requiring transformation.
18 The conversation controller uses transformations that are put into a common registry by a
19 service. In alternative embodiments, more than one conversation controller may be used.

20 A document handler may include deployment descriptors containing mappings
21 and/or transformations that are applied to a document, such that the documents may be
22 used by the existing business logic of the web service provider. It will be apparent to one
23 of ordinary skill in the art that the deployment descriptors may be coded by a TCDL
24 programmer for the web service provider depending on the existing code.

25 **Description Of Drawings**

26 An exemplary embodiment of the invention is illustrated in the drawings in which
27 like numeral references refer to like elements, and wherein:

28 FIGURE 1 is a diagram illustrating a comparison of two E-service conversations
29 for similar services;

30 FIGURE 2 is a block diagram of an exemplary embodiment of a web service
31 provider employing the principles of an embodiment of the present invention;

32 FIGURE 3 is a block diagram of an exemplary system employing the principles of
33 an embodiment of the invention; and

1 FIGURE 4 is an exemplary XSLT stylesheet showing the transformation required
2 to compute an *extended_price* according to an embodiment of the invention.

3 **Detailed Description**

4 The numerous innovative teachings of the present application will be described
5 with particular reference to the presently preferred exemplary embodiments. However, it
6 should be understood that this class of embodiments provides only a few examples of the
7 many advantageous uses of the innovative teachings herein. In general, statements made
8 in the specification of the present application do not necessarily delimit any of the various
9 claimed inventions. Moreover, some statements may apply to some inventive features but
10 not to others.

11 There are three elements to a basic WSCL (Web Services Conversation
12 Language) specification:

- 13 1. **Document type descriptions** specify the types (schemas) of XML
14 documents that the service can accept and transmit in the course of a
15 conversation.
- 16 2. **Interactions** model the states of the conversation as document exchanges
17 between conversation participants. WSCL currently supports four types
18 of interactions: *Send* (the service sends out an outbound document),
19 *Receive* (the service receives an inbound document), *SendReceive* (the
20 service sends out an outbound document, then expects to receive an
21 inbound document in reply), and *ReceiveSend* (the service receives an
22 inbound document and then sends out an outbound document).
- 23 3. **Transitions** specify the ordering relationships between interactions. A
24 transition specifies a source interaction, a destination interaction, and a
25 document type that triggers the transition. WSCL 1.0 also supports two
26 special transitions: *Default Transition* and *Exception Transition*. A
27 default transition is triggered if a valid inbound (for a *SendReceive*
28 interaction) or outbound (for a *ReceiveSend* interaction) document is
29 received for a given interaction, but no other transition is triggered. At
30 most one default transition can be defined per source interaction.

31 In an exemplary embodiment of the present invention, WSCL is extended with a
32 new element, *Transformations*, that specifies the transformations that can be applied to
33 the documents used in the conversation specification. The WSCL *Interaction* element is

1 then extended with a “Transformations” subelement that maps the inbound and outbound
2 documents to these transformations. This extended language is referred to as
3 “Transformational Conversation Definition Language” (TCDL).

4 Extending conversation specifications with document transformations frees
5 service developers from having to navigate between conflicting document types. This
6 completely decouples services from any awareness of document type mismatches – such
7 mismatches can be compensated for in the conversation specification.

8 In addition, this enables a third-party conversation controller to assume
9 responsibility for some of the computational tasks that may be involved in the course of
10 the conversation. For example, the conversation controller could use these
11 transformations to route a conversation according to the content (vs. type) of message
12 documents. A preferred conversation controller is described in U.S. Patent Application
13 Ser. No. 09/862,612, (Attorney Docket No. HP 10012649-1), entitled “Lightweight
14 Dynamic Service Conversation Controller,” to Michael J. Lemon, et al., filed on May 23,
15 2001, and herein incorporated by reference in its entirety. It will be apparent to one
16 skilled in the art how to extend the preferred conversation controller, or alternative
17 conversation controller, to enable document transformation based on the description of
18 the present invention herein.

19 Referring now to the drawings, and in particular to FIGURE 1, there are shown
20 two “shopping cart” conversation specifications 101 and 151 for digital photograph
21 storage services (similar to services offered by companies such as ofoto.com). These
22 conversations are specified from the perspective of the photo storage service. The circles
23 represent interactions, or states; the boxes represent inbound document types, and the arcs
24 between the circles represent the transitions between interactions, which are driven by
25 outbound document types. The two conversations 101 and 151 are structurally similar,
26 but not identical.

27 The “secure album” conversation 101 requires the client to sign in (login) before
28 selecting an album, as shown by the outbound document type *loginRS* 103 becoming an
29 inbound document along with *chooseAlbum* 105 required for the album selection
30 transaction 107. On the other hand, the “anonymous guest” conversation 151 doesn’t
31 require the client to sign in (login) until they are ready to purchase some photos, as shown
32 by inbound document *chooseAlbum* 155 to the start transaction 153 which does not
33 require a *loginRS* 103 document. In addition, once the client is ready to check out, the
34 “secure album” conversation expects the client to send a document of type *CheckoutRQ*

1 109 and the server to respond with a document of type *Bill* 111, while the “anonymous
2 guest” conversation expects the client to send a *RequestInvoice* 159 document and the
3 service to respond with an *Invoice* 161 document. Thus, it can be seen that these two
4 similar conversations produce either a *bill* or *invoice*, respectively, but that the documents
5 are not identical.

6 For purposes of this description, a service is loosely defined as an application that
7 can be discovered and that supports some protocol, so that another application could send
8 it a message. Thus, there is a discovery mechanism and a protocol. Not necessary for the
9 present invention, but in an optimal e-service marketplace, a service is modular and self-
10 contained; services or applications can act as resources to other applications so that they
11 can be performed or scheduled or committed with or without human intervention.

12 Another optional requirement for a service is that it be a self describing application. In
13 this way, it can communicate its capabilities and requirements to other applications
14 through a pre-defined or standard mechanism or protocol. A service is instrumented so
15 that an external application management system is able to detect and manage the state of
16 the service and the status of its outcome. Further, one should be able to broker auction
17 services, meaning that there should be a marketplace that exists. It will be apparent to
18 one skilled in the art that for the present invention, only the most rudimentary service
19 characteristics must be implemented. It will also be apparent to one skilled in the art that
20 as E-service technology evolves, additional service characteristics, as listed above, will be
21 available.

22 Assuming a marketplace with services that are capable of exchanging messages,
23 service implementers/designers must solve three problems in order to enable the services
24 to dynamically interact with each other without a human directing them. First, new
25 services must be capable of being located so services can decide to talk to each other.
26 Second, services must be able to send messages to each other, that is legal, or allowable,
27 messages. This is referred to as a message exchange. For example, service A could
28 accept a login and a login message with the understanding that it will return to the client
29 either a login-accepted or a login-rejected message. This is a message exchange. Third,
30 services must be able to automatically transmit and accept messages in a variety of
31 formats because each service may have its own document format.

32 A conversation is a sequence of message exchanges. For example, the client
33 sends a login document to Service A. Service A sends a login accepted document to the
34 client. The client then sends a catalog request to Service A. Service A then sends a

1 catalog to the client, etc. A conversation description is a specification that gives a formal
2 description, or formal definition, of a set of legal sequences of document exchanges. This
3 present invention extends a conversation exchange to include document transformation.

4 The messages exchanged by services are assumed to be semi-structured
5 documents, for example using XML. The structure for a class of documents is described
6 using a schema or some other specification. Because the documents are structured or
7 semi-structured, a transformation from one type of schema to another type of schema can
8 be written. In an exemplary embodiment, a transformation is necessary to transform a
9 document of type *login_request* to type *sign_in_request*. The difference between the two
10 documents might be very simple. For example, the *login_request* document might expect
11 a “login” field, whereas a *sign_in_request* expects a “user_name” field or the similar.
12 Message and communication protocols typically require header information to be sent
13 along with the data. Specifically, XML messages basically have a header that identifies
14 the type of the object that is being sent. Therefore, even though in an abstract level, the
15 “login” and “user_name” fields hold the same data, the messages will be sent with
16 headers that label the data differently. Thus, even if the data is identical, without an
17 appropriate transformation, the message will be rejected if it contains an alternative
18 header.

19 A model for web communication is that web services publish information about
20 the specification that they support. UDDI (Universal Description Discovery and
21 Integration) facilitates the publication and discovery of web service information. A
22 current version of WSDL (Web Service Definition Language 1.0) is an XML-based
23 format that describes the interfaces and protocol bindings of web service functional
24 endpoints. WSDL also defines the payload that is exchanged using a specific messaging
25 protocol; Simple Object Access Protocol (SOAP) is one such possible protocol.

26 A Conversation Design Language (CDL), as described in U.S. Provisional Patent
27 Application Serial No. 60/253,953, (Attorney Docket No. HP 10010485-1), entitled “A
28 Computer Language for Defining Business Conversations,” to Alan Karp, et al., filed on
29 Nov. 28, 2000, and herein incorporated by reference in its entirety, enables web services
30 provided by different entities to engage in flexible and autonomous interactions. For
31 example, supposing a client web service in one business discovers a storefront web
32 service provided by another business. These services can communicate by exchanging
33 messages using a common transport (e.g., HTTP) and message format (e.g., SOAP).
34 However, also suppose that the storefront service expects the message exchanges to

1 follow a specific pattern (conversation). CDL may be used to define the conversation,
2 such that the storefront service may expect a particular message in response to
3 transmitting a particular message.

4 Messages exchanged between web services may include XML documents.
5 Messages may include different types of messages (*e.g.*, types of XML documents), and a
6 type of message may be described by a schema (*e.g.*, in a registry) and facilitates the
7 introspection of services and their interfaces. It will be apparent to one of ordinary skill
8 in the art that messages may be generated using languages other than XML.

9 An exemplary conversation includes a sequence of exchanges of XML documents
10 between entities. A TCDL description file includes the sequence of interactions (*e.g.*,
11 transmitting and/or receiving messages) between entities and the XML document types
12 that may be used in each interaction. Each TCDL description describes a single type of
13 conversation from the perspective of a single participant. A service can participate in
14 multiple types of conversations. Furthermore, a service can engage in multiple
15 simultaneous instances of different conversations. A TCDL programmer may create a
16 TCDL description for a conversation , and publish it in a UDDI-like registry. A
17 developer who wants to create a service that supported a conversation creates and
18 documents service endpoints that support the messages specified by the TCDL
19 description for that conversation.

20 FIGURE 2 illustrates an exemplary embodiment of a web service provider 200
21 employing principles of an embodiment of the invention. The web service provider 200
22 may publish a TCDL description file 210 in a remote registry, such as registry 310 (as
23 shown in FIGURE 3), storing TCDL description files defining conversations for multiple
24 service providers. Other web service providers and customers may retrieve the TCDL
25 description file 210 from the registry, because the TCDL description file 210 defines a
26 conversation for interacting with the web service provider 200 to facilitate business-to-
27 business transactions and customer-to-business transactions with the web service provider
28 200. Further, the TCDL description file defines valid document transformations and
29 identifies locations of appropriate schema documents.

30 Instead of, or in addition to TCDL description files, the registry 310 may include a
31 list of CDL description files used by each service provider, *e.g.*, a TCDL description file
32 without any transformations defined. For example, a user may store a plurality of
33 CDL/TCDL description files for communicating with multiple web service providers.
34 The user may access the registry 310 to identify the CDL/TCDL description file used by a

1 specific web service provider. Then the user, already having the identified CDL/TCDL
2 description file, uses the identified CDL/TCDL file to communicate with the web service
3 provider. Furthermore, the user may not need to access the registry if the user knows
4 which CDL/TCDL description files is used by this service provider. This allows a
5 service provider to communicate with other services using transformation if a TCDL file
6 is available, but to also communicate with a service provider without transformation if
7 only a CDL file is available and inbound and outbound documents are valid without
8 transformation.

9 In one embodiment, the web service provider 200 compiles the TCDL description
10 file 210 to generate a conversation controller 220 (e.g., an executable computer program).
11 The conversation controller 220 may transmit and receive XML documents 230 and
12 generate error messages based on the TCDL description file 210. A document handler
13 240 may include deployment descriptors containing mappings and/or transformations that
14 are applied to the XML document 230, such that the XML documents 230 may be used
15 by the existing business logic 250 of the web service provider 200. It will be apparent to
16 one of ordinary skill in the art that the deployment descriptors may be coded by a TCDL
17 programmer for the web service provider 200 depending on the existing code.

18 The TCDL description file 210 may also be incorporated in a service interface,
19 rather than being compiled and executed as a separate program. For example, the TCDL
20 description file 210 may be a library or object.

21 FIGURE 3 illustrates an exemplary system 300 employing principles of an
22 embodiment of the invention. The system 300 embodies an example where a customer
23 340 orders items from an entity A (e.g., a web service provider) providing a web service
24 on a computer 320. The entity A sends the order, originating from the customer 340, to a
25 supplier (i.e., entity B) providing a sales order web service using a computer 330. The
26 web services provided by entities A and B may be configured similarly to the web service
27 shown in FIGURE 2.

28 The entities A and B may be registered in registry 310. The registry 310 may
29 include a UDDI-like registry that lists a location, which may include a uniform resource
30 identifier (e.g., URL, URN, and the like) for each CDL/TCDL description file provided
31 by the entities A and B. In addition to or alternatively, the registry 310 may include a list
32 of CDL/TCDL description files used by entities A and B. Therefore, a user or web
33 service provider that desires to communicate with entity A or entity B may access the
34 registry to determine which CDL/TCDL description file is used by the entity. The

1 registry 310 may be accessed through a global computer network, *i.e.*, the Internet, and or
2 through private networks, *i.e.*, intranets or extranets. Entities, including customers,
3 businesses, and the like, may access the registry 310 to identify web services provided by
4 each other. For example, the customer 340 may identify the purchase order service
5 provided by the entity A, and entity A may identify the sales order service provided by
6 the entity B through the registry 310.

7 The customer 340 may utilize a computer 345 executing a TCDL client 347 that
8 exchanges documents with a purchase order conversation controller 325 in the computer
9 320. The TCDL client 347 may retrieve the URL (*e.g.*, <http://www.entityA.com/po>) of
10 the purchase order conversation controller 325. The interactions between customer 340
11 and the entity A may include transmit/receive purchase order, receive invoice, transmit
12 payment, and the like.

13 The entity A may retrieve the URL (*e.g.*, <http://www.entityB.com/sales>) for the
14 sales order controller 332. Entity A and the entity B may then engage in a conversation to
15 facilitate a purchase from the entity B. The interactions between entity A and the entity B
16 may include transmit price proposed, receive price accepted, receive price rejected,
17 receive invoice, transmit receipt, and the like.

18 The computer 320 and the computer 330 may include web servers providing a
19 service over the Internet , and the computer 345 may include a conventional device
20 configured to communicate over the Internet and/or other networks. It will be apparent to
21 one of ordinary skill in the art that the system 300 is functional to provide other services
22 to one or more entities, which may include one or more customers, businesses, and the
23 like.

24 Web services communicate with each other through the exchange of documents,
25 and a TCDL conversation description defines all the inbound and outbound document
26 types that may be used in the conversation using document type descriptions, as well as
27 interactions, transformations and transitions. A document type description may refer to a
28 schema (*e.g.*, an XML schema, and the like) that includes attributes (*e.g.*, data types, and
29 the like) of a particular document type. A document (*e.g.*, an XML document, and the
30 like) of a particular document type includes an instance of the attributes included in the
31 schema for that document type. The schemas of the documents exchanged during a
32 conversation are not specified as part of the CDL specification. The actual document
33 schemas may be defined in XML documents that can be referenced by their URL or URN
34 in the interaction elements of the conversation specification. For example, the following

1 document type description defines an input document (*i.e.*, inbound XML document) that
2 conforms to a purchase order schema defined in a file named *billOfSales.xsd*, where the
3 extension “xsd” indicates an XML schema document.

```
4 <InboundXMLDocuments>
5     <InboundXMLDocument
6         hrefSchema="http://conv123.org/billOfSales.xsd"
7         id="billOfSales">
8     </InboundXMLDocument>
9 </InboundXMLDocuments>
```

10 The following document type description defines an output document (*i.e.*,
11 outbound XML document) that conforms to a purchase order schema defined in a file
12 named *paymentDetails.xsd*.

```
13 <OutboundXMLDocuments>
14     <OutboundXMLDocument
15         hrefSchema="http://conv123.org/paymentDetails.xsd"
16         id="paymentDetails">
17     </OutboundXMLDocument>
18 </OutboundXMLDocuments>
19     00
```

20 Suppose that entity A uses documents of the type *billOfSales*, but that entity B
21 uses a similar document type of *paymentRequest*. Without the ability to transform one
22 document type into the other, one cannot communicate autonomously between the two
23 entities. The present invention extends the document type description to include a
24 transformations attribute.

25 Suppose there were an XSLT stylesheet, *paymentRequest2billOfSales*, that could
26 transform a *paymentRequest* document into the document type *billOfSales*, and another
27 stylesheet, *paymentResponse2paymentDetails* that could transform a *paymentResponse*
28 document into type *paymentDetails*. The following XML code is a TCDL representation
29 of the *Transformations* element of the specification for Conversation 151 (as shown in
30 FIGURE 1). It will be apparent to one of ordinary skill in the art that TCDL is not tied to
31 any particular transformation specification. XSLT is used for exemplary purposes.

```
32 <Transformations>
33     <Transformation id="paymentRequest2billOfSales">
34         <href="http://conv123.org/paymentRequest2billOfSales.xsl"/>
35         <InboundXMLDocument
36             hrefSchema="http://conv123.org/paymentRequest.xsd"/>
37         <OutboundXMLDocument
```

1 hrefSchema="http://conv123.org/billOfSales.xsd"/> >
2 </Transformation>
3 <Transformation id="paymentResponse2paymentDetails">
4 <href="http://conv123.org/ paymentResponse2paymentDetails.xsl"/>
5 <InboundXMLDocument
6 hrefSchema="http://conv123.org/paymentResponse.xsd"/>
7 <OutboundXMLDocument
8 hrefSchema="http://conv123.org/paymentDetails.xsd"/>
9 </Transformation>
10 </Transformations>

11 The TCDL above describes the ability for a service to transform a check out
12 request into a request invoice, or in the terminology of the services, a “paymentRequest”
13 to a “billOfSales”. The URL for the transformation XSLT stylesheet is defined in
14 <href="http://conv123.org/paymentRequest2billOfSales.xsl"/>, as shown in
15 the transformation id attribute. The transformation is associated with an inbound
16 document

17 <InboundXMLDocument hrefSchema="http://conv123.org/paymentRequest.xsd"/>
18 which is transformed into an outbound document
19 <OutboundXMLDocument hrefSchema="http://conv123.org/billOfSales.xsd"/>.

20 The TCDL above also describes the ability for a service to transform a payment
21 response into a payment details document. The URL for the transformation XSLT
22 stylesheet is defined in
23 <href="http://conv123.org/paymentResponse2paymentDetails.xsl"/>, as shown
24 in the transformation id attribute. The transformation is associated with an inbound
25 document

26 <InboundXMLDocument hrefSchema="http://conv123.org/paymentResponse.xsd"/>
27 which is transformed into an outbound document
28 <OutboundXMLDocument hrefSchema="http://conv123.org/paymentDetails.xsd"/>.

29 The following is an instance of using TCDL to couple the transformation with the
30 interaction. Interaction describes the message exchange. Inbound XML document
31 describes the message that is coming in and outbound describes the message that is going
32 out. The interaction section of the TCDL defines, for a particular state, that an input bill
33 of sales document is expected to be received. It expects to then put as output a payment
34 details document. The extension is the transformation element which defines the
35 application of the payment request to bill of sales transformation to an inbound document.
36 Similarly the payment response to payment details transformation can be applied to the

1 outgoing document, if necessary. It will be apparent to one of ordinary skill in the art that
2 while the exemplary embodiment shows one transformation for one type of inbound
3 document, e.g., *billOfSales*, and one type of outbound document, e.g., *paymentDetails*,
4 that many transformations could be defined for any of several inbound or outbound
5 documents. These transformations are added to an exemplary extended TCDL
6 interaction, as shown below.

```
7 <Interaction StepType="ReceiveSend" id="payment" initialStep="false">
8     <InboundXMLDocuments>
9         <InboundXMLDocument
10            hrefSchema="http://conv123.org/billOfSales.xsd"
11            id="billOfSales">
12            <Transformations>
13                <Transformation id="paymentRequest2billOfSales"/>
14            </Transformations>
15        </InboundXMLDocument>
16    </InboundXMLDocuments>
17    <OutboundXMLDocuments>
18        <OutboundXMLDocument
19            hrefSchema="http://conv123.org/paymentDetails.xsd"
20            id="paymentDetails">
21            <Transformations>
22                <Transformation id="paymentResponse2paymentDetails"/>
23            </Transformations>
24        </OutboundXMLDocument>
25    </OutboundXMLDocuments>
26 </Interaction>
```

27 For each state a piece of XML code (TCDL) is provided that describes the schema
28 that describes the data and the transformations that could be used to change the document
29 format used by one service to the document format of another service. TCDL specifies
30 the valid inbound and outbound documents for an interaction, it does not specify how the
31 conversation participants will handle and produce these documents. The TCDL
32 specification of a conversation is thus service-independent, and can be used (and reused)
33 by any number of services. Further, the ordering among interactions is specified in a
34 *transitions* section of the TCDL, and is not relevant to the transformation definitions
35 because regardless of the form of conversation navigated to reach a state, or interaction,
36 that interaction is defined to utilize a specific set of inbound and outbound documents,
37 each of a given format.

1 In another exemplary embodiment, a client shops for grocery items using a
2 service. An XML description for a shopping list document for Service A might look like
3 the following:

```
4 <shopping_list>
5   <grocery>
6     <product> "bread" </product>
7     <qty> 1 </qty>
8     <unit_price> 1.75 </unit_price>
9   </grocery>
10  <grocery>
11    <product> "bacon" </product>
12    <qty> 1 </qty>
13    <unit_price> 6.00 </unit_price>
14  </grocery>
15  <grocery>
16    <product> "lettuce" </product>
17    <qty> 2 </qty>
18    <unit_price> 1.25 </unit_price>
19  </grocery>
20  <grocery>
21    <product> "tomato" </product>
22    <qty> 3 </qty>
23    <unit_price> 0.45 </unit_price>
24  </grocery>
25 </shopping_list>
```

26 In contrast, Service B may use an extended shopping list document like the
27 following:

```
28 <shopping_list>
29   <grocery>
30     <product>"bread"</product>
31     <qty>1</qty>
32     <unit_price>1.75</unit_price>
33     <extended_price>1.75</extended_price>
34   </grocery>
35   <grocery>
36     <product>"bacon"</product>
37     <qty>1</qty>
38     <unit_price>6.00</unit_price>
39     <extended_price>6</extended_price>
40   </grocery>
```

```
1      <grocery>
2          <product>"lettuce"</product>
3          <qty>2</qty>
4          <unit_price>1.25</unit_price>
5          <extended_price>2.5</extended_price>
6      </grocery>
7      <grocery>
8          <product>"tomato"</product>
9          <qty>3</qty>
10         <unit_price>0.45</unit_price>
11         <extended_price>1.35</extended_price>
12     </grocery>
13 </shopping_list>
```

14 The extended shopping list includes a field *extended_price* which is merely a
15 calculation of quantity (*e.g.*, *qty*) and the unit price (*e.g.*, *unit_price*). An appropriate
16 transformation is required if Service A and Service B want to communicate in a
17 conversation. Further, including a transformation in a stylesheet used by the conversation
18 controller allows any computational loads to be transferred to the conversation controller
19 rather than the client site or service provider site. This may be advantageous for
20 transformations with significant computational needs.

21 FIGURE 4 shows an exemplary XSLT stylesheet 400 showing the transformation
22 required to compute *extended_price*. This exemplary stylesheet 400 has three templates:
23 *shopping_list* 401, *grocery* 403, and a common template for *product*, *qty* and *unit_price*
24 405. The *shopping_list* template 401 applies templates for *grocery* 413. The *grocery*
25 template 403 applies templates for *product*, *qty* and *unit_price* 433, as necessary. The
26 element *extended_price* 435 is also defined for template *grocery* 403. The
27 *extended_price* element is defined to perform the calculation *unit_price* * *qty*. This
28 stylesheet is used to transform the shopping list above to the extended shopping list.

29 In an exemplary embodiment a *log_in* document must be transformed into a
30 *sign_in* document. A shopper sends a *log_in* request to a service provider. The
31 conversation conducted between the shopper and the service provider is managed by a
32 conversation controller/server. The conversation controller determines that the service
33 provider needs a *sign_in* document and that a transformation is defined in the TCDL
34 document corresponding to transformation of a *log_in* to a *sign_in* document. This
35 transformation definition points to an XML schema. The conversation controller
36 translates the *log_in* request sent by the shopper to a *sign_in* request using the XML

1 schema for the transformation. The conversation controller then forwards the sign_in
2 request to the service provider. This entire process is transparent to the user, who is
3 unaware that any document transformation has taken place.

4 Having described preferred embodiments of a novel method for extending a
5 conversation definition language to include document transformation capability (which
6 are intended to be illustrative and not limiting), it is noted that modifications and
7 variations can be made by persons skilled in the art in light of the above teachings. It is
8 therefore to be understood that changes may be made in the particular embodiments of the
9 invention disclosed which are within the scope and spirit of the invention as defined by
10 the appended claims.